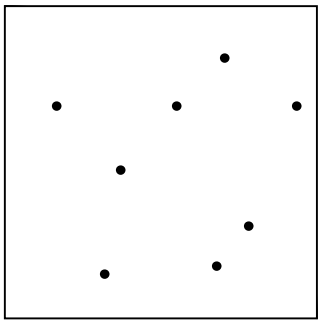
Prof. Dr. Andreas Künkler

Programmiersprachen (C++)

Übung 2
Ausdrücke und Anweisungen

Aufgabe 1
Schreiben Sie ein C++-Programm, das die Summe und den Mittelwert einer Anzahl von positiven ganzen Zahlen berechnet. Lesen Sie die Zahlen in einer Schleife nach und nach von der Tastatur ein und führen Sie dabei gleich die nötige Verarbeitung durch. Brechen Sie die Eingabeschleife ab, sobald eine negative Zahl eingegeben wurde.

Aufgabe 2
In dieser Aufgabe geht es darum, die Zahl π genähert zu berechnen. Um π zu ermitteln, genügt es, die Fläche des Viertels des Einheitskreises zu berechnen. Da die Fläche des Einheitskreises gleich π ist, muss man nur die ermittelte Fläche des Viertelkreises mit 4 multiplizieren um auf π zu kommen. Diese Idee ist in der folgenden Abbildung dargestellt. Man bestimmt zufällige Punkte in einem Einheitsquadrat. Die Fläche unter dem Viertelkreis ist dann ungefähr gleich der Anzahl der Punkte, die unter der Kreislinie liegen, dividiert durch die Gesamtzahl der Punkte. Dieses Verfahren zur Ermittlung einer Fläche (allgemeiner: eines bestimmten Integrals) heißt wegen seines Zufallscharakters auch Monte-Carlo-Integration.



Schreiben Sie ein Programm, das mittels Monte-Carlo-Integration einen Näherungswert für π berechnet. Lesen Sie dazu eine ganze Zahl ein, die angibt, wie viele Punkte „gewürfelt“ werden sollen. Würfeln Sie entsprechend oft und berechnen Sie den Quotienten aus der Anzahl der Punkte unter der Kreislinie durch die Gesamtzahl der Punkte. Ein Punkt (x, y) liegt unter der Kreislinie, wenn $x^2 + y^2 \leq 1.0$ gilt.

Zur Initialisierung des Pseudo-Zufallszahlen-Generators können Sie die Funktionen `srand()` aus dem Header `<cstdlib>` und `time()` aus dem Header `<ctime>` verwenden:

```
srand(time(NULL) * time(NULL));
```

Eine im Intervall [0.0 , 1.0] liegende reelle Zufallszahl erhalten Sie z.B. durch

```
double x = double(rand())/double(RAND_MAX);
```

Aufgabe 3
Schreiben Sie ein Programm zur Primzahlbestimmung nach der Methode „Sieb des Eratosthenes“. Die Grundidee besteht darin, dass in dem Programm ein boolesches Feld enthalten ist, in dem für jede Zahl die Information gespeichert ist, ob es sich um eine Primzahl handelt oder nicht. Zunächst werden alle Feldelemente auf `true` gesetzt. Als nächstes formulieren Sie eine Schleife, die beginnend mit der Zahl 2 von Primzahl zu Primzahl springt. Sobald (in dem Feld) eine Primzahl entdeckt wurde, werden die Feldinhalte für alle Vielfachen dieser Zahl auf `false` gesetzt. Das Programm wird also zunächst die Feldelemente mit den geraden Indizes auf `false` setzen, dann 3 als nächste Primzahl „entdecken“, die Vielfachen von 3 mit `false` markieren, die 4 (die jetzt auf `false` steht) überspringen usw.

Aufgabe 4
Um festzustellen, ob Ihr Rechner auch wirklich den ASCII-Zeichensatz verwendet, sollen die druckbaren Zeichen (ASCII-Code 0x20 ... 0x7e) als Tabelle ausgeben werden. Anstelle der nicht druckbaren Zeichen wird ein Punkt als Ersatzzeichen ausgegeben. Schreiben Sie dazu ein Programm, das die folgende Ausgabe erzeugt:

```
   0  1  2  3  4  5  6  7
0  .  .     0  @  P  `  p
1  .  .  !  1  A  Q  a  q
2  .  .  "  2  B  R  b  r
3  .  .  #  3  C  S  c  s
4  .  .  $  4  D  T  d  t
5  .  .  %  5  E  U  e  u
6  .  .  &  6  F  V  f  v
7  .  .  '  7  G  W  g  w
8  .  .  (  8  H  X  h  x
9  .  .  )  9  I  Y  i  y
a  .  .  *  :  J  Z  j  z
b  .  .  +  ;  K  [  k  {
c  .  .  ,  <  L  \  l  |
d  .  .  -  =  M  ]  m  }
e  .  .  .  >  N  ^  n  ~
f  .  .  /  ?  O  _  o  .
```

Aufgabe 5
Die Dateiverzeichnisse

```
V:\kuenkler\CPP\vorlesung\beispiele\kap04
V:\kuenkler\CPP\vorlesung\beispiele\kap05
```

enthalten C++-Quelldateien. Übersetzen Sie diese Dateien und starten Sie die so entstandenen C++-Programme. Versuchen Sie nachzuvollziehen, wie die Programme arbeiten..